A hybrid OpenMP and MPI implementation of a conservative spectral method for the Boltzmann equation

Jeffrey R. Haack *

January 18, 2013

Abstract

We demonstrate the implementation of a hybrid OpenMP and MPI parallelization of a conservative spectral method for the Boltzmann equation originally developed by Gamba and Tharkabhushaman. We perform a scaling analysis to demonstrate that the problem is well suited to parallelization, and find that the computational time scales linearly with the number of compute nodes on high performance computing resources. The original method has also been improved to higher order in space and time and is implemented on non-uniform grids in physical space. We test this scheme for an example problem in which a kinetic boundary layer generates a shock wave for large space and long times. This is the first time that the fully nonlinear Boltzmann collision operator has been used to compute this problem.

1 Introduction

In this paper we consider the solution of the fully nonlinear space inhomogenous Boltzmann equation, which is an integrodifferential equation that describes the evolution of a dilute gas where only binary interactions between particles are considered. In particular, we extend the conservative spectral method developed by Gamba and Tharkabhushanam [21, 22] for use on high performance computing resources using OpenMP and MPI, and study the evolution of a kinetic boundary layer problem that was originally computed by Aoki et al [3] using the Bhatnagar-Gross-Krook (BGK) approximation for collisions. There are many difficulties associated with numerically solving the Boltzmann equation, most notably the dimensionality of the problem and the conservation of the collision invariants. For physically relevant three dimensional applications the distribution function is seven dimensional and the velocity domain is unbounded. In

*Department of Mathematics, The University of Texas at Austin, 2515 Speedway, Stop C1200 Austin, Texas 78712

addition, the collision operator is nonlinear and requires evaluation of a five dimensional integral at each point in phase space. The collision operator also locally conserves mass, momentum, and energy, and any approximation must maintains this property to ensure that macroscopic quantities evolve correctly.

Most numerical computation of the collision operator is based on stochastic Monte Carlo methods, for example the methods of Bird [5] and Nanbu [31]. These methods strongly reduce the dimensionality of the problem by approximating the collision mechanism through a stochastic, particle-based description, which ensures that time and memory is not wasted on computing regions where f is near zero. Because these methods use the microscopic collision mechanism directly they exactly conserve the quantities such as mass or energy. However, the stochastic nature of Monte Carlo methods give rise to statistical fluctuations in the numerical results. The amount of computational resources required grow very quickly in nonsteady problems, as well as problems where collisions strongly dominate the system, flows with high mean velocities, and problems with a distribution function that is far from equilibrium. In recent years deterministic methods have been able to catch up with Monte Carlo solvers, obtaining similar results with relatively coarse meshes and providing a faster rate of convergence without having to worry about the fluctutations or how close they are to Maxwellians [13].

To avoid these problems, a deterministic class of so called discrete velocity methods (DVM) were proposed. Rather than using a random collection of particles to collide, these methods discretize the velocity space on a finite grid in such a way that the collision mechanics are satisfied exactly for pairs of velocity grid points. This method originated with Broadwell [10], and was later extended by others [12, 11, 25, 36]. This approach requires careful choice of grid points in velocity space and on the unit sphere to preserve the conserved quantities. However, while these methods are able to satisfy the conservation properties of the collision operator, many have observed accuracy of only $O(N^{-d/2})$, where N is the total number of velocity grid points in each dimension, despite being as computationally expensive as naively applying a standard quadrature formula directly to the collisional integral without any regard for conservation [7, 33]. Recent work by the group of Varghese [30] uses a combined Monte-Carlo and discrete velocity formulation, which selects random pairs of velocity grid points for collision, then conservatively interpolates the result back onto the grid. This method is less noisy than classical DSMC, and outperforms it in regions where there is less activity.

Spectral methods are a deterministic approach that compute the collision operator to high accuracy by exploiting its Fourier structure. These methods grew from the analytical works of Bobylev [6] developed for the Boltzmann equation for Maxwell type potential interactions and integrable angular cross section, where the corresponding Fourier transformed equation takes a closed form. Spectral methods provide many advantages over Direct Simulation Monte Carlo Methods (DSMC) because they are more suited to time dependent problems, low Mach number flows, high mean velocity flows, and flows that are away from equilibrium. In addition, deterministic methods avoid the statistical fluc-

tuations that are typical of particle based methods. Spectral approximations for this type of models where first proposed by Pareschi and Perthame [34]. Later Pareschi and Russo [35] applied this work to variable hard potentials by periodizing the problem and its solution and implementing spectral collocation methods.

Using this representation, Bobylev and Rjasanow developed methods for the case of Maxwell molecules [8] and hard spheres [9] to derive the convolution and approximate it with numerical quadrature. Ibragimov and Rjasanow extended these ideas to the more general case of variable hard spheres [24] and simplified the integration domain by explicitly computing the spherical integral in the derivation of the weight function for the convolution. In a related work, Pareschi and Russo [35] applied this framework to develop methods for the Maxwell molecules, hard spheres, and variable hard spheres cases using a collocation method, which uses orthogonal polynomials to reduce the convolution integral to a convolution sum, providing spectral accuracy to the collision integral computation.

Inspired by the work of Ibragimov and Rjasanow [24], Gamba and Tharkabhushanam [21, 22] observed that the Fourier transformed collision operator takes a simple form of a weighted convolution and developed a spectral method based on the weak form of the Boltzmann equation that provides a general framework for computing both elastic and inelastic collisions. Macroscopic conservation is enforced by solving a numerical constrained optimization problem that finds the closest distribution function in L_2 to the output of the collision term that conserves the macroscopic quantities. These methods do not rely on periodization but rather on the use of the fast Fourier transform in the computational domain, and convergence to the solution of the continuous problem is obtained by the use of the extension operator in Sobolev spaces [2].

All of these methods are computationally expensive, so there is a natural desire to extend them to high performance computing resources. However, there have been relatively few published works on parallelization of Boltzmann solvers. Graphics Processing Units (GPUs) have emerged as a resource for applications other than graphics, providing hundreds to thousands of lightweight independent processors already geared to independently operating on large sets of data. Frezzotti and collaborators [16, 15, 14] implemented DSMC and BGK type Boltzmann models on GPUs, and in parallel Malkov and Ivanov and collaborators have explored implementing classical DSMC as well as the discrete velocity Monte Carlo of Varghese et al. on GPUs [28, 29, 26]. Many existing codes cannot be ported directly to GPUs, due to their relatively simple instruction set and complicated memory management structure. In 2012, Intel released a new architecture known as Many Integrated Core (MIC). This architecture combines many more powerful cores on a single chip than ever before, all able to access the same pooled memory. These cores are classical CPUs, rather than GPUs, which allows relatively straightforward speedup of existing codes [19].

This paper presents the extension of the deterministic spectral method of Gamba and Tharkabhushanam to CPU-based parallelization using the OpenMP and MPI APIs, as opposed to GPUs. We also present a second order in time

and space extension of this method that implements nonuniform grids. Sections 2 and 3 are presented for the benefit of the reader; much of the spectral formulation can be found in [21, 22]. This method has been tested on the Ranger supercomputer at the Texas Advanced Computing Center (TACC) and timing studies are provided to explore the scaling of the method to more and more processors for large problems, and will serve as a stepping stone to future implementation on MIC architecture. We present some $1D$ in physical space results for a problem proposed by Aoki et al.[3] where a sudden change in wall temperature results in a shock that cannot be explained by classical hydrodynamics.

2 The space inhomogeneous Boltzmann equation

The space inhomogeneous initial-boundary value problem for the Boltzmann equation is given by

$$\frac{d}{dt}f(\mathbf{x},\mathbf{v},t) + \mathbf{v}\cdot\nabla_x f(\mathbf{x},\mathbf{v},t) = \frac{1}{\varepsilon}Q(f,f), \tag{1}$$

with

$$\begin{aligned} &\mathbf{x}\in\Omega\subset\mathbb{R}^d, \qquad \mathbf{v}\in\mathbb{R}^d \\ &f(\mathbf{x},\mathbf{v},0) = f_0(\mathbf{x},\mathbf{v}) \\ &f(\mathbf{x},\mathbf{v},t) = f_B(\mathbf{x},\mathbf{v},t), \qquad \mathbf{x}\ \in\partial\Omega. \end{aligned}$$

where $f(\mathbf{v},t)$ is a probability density distribution in $\mathbf{v}$-space and f_0 is assumed to be locally integrable with respect to $\mathbf{v}$ and the spatial boundary condition f_B will be specified in below. The dimensionless parameter $\varepsilon > 0$ is the scaled Knudsen number, which is defined as the ratio between the mean free path between collisions and a reference macroscopic length scale.

The collision operator $Q(f,f)(\mathbf{x},\mathbf{v},t)$ is a bilinear integral form local in t and $\mathbf{x}$ and is given by

$$Q(f,f)(\cdot,\mathbf{v},\cdot) = \int_{\mathbf{v}_*\in\mathbb{R}^d}\int_{\sigma\in S^{d-1}} B(|\mathbf{v}-\mathbf{v}_*|,\cos\theta)(f(\mathbf{v}'_*)f(\mathbf{v}')-f(\mathbf{v}_*)f(\mathbf{v}))d\sigma d\mathbf{v}_*, \tag{2}$$

where the velocities $\mathbf{v}', \mathbf{v}'_*$ are determined through a collision rule (3), depending on $\mathbf{v}, \mathbf{v}_*,$, and the positive term of the integral in (2) evaluates f in the pre-collisional velocities that will take the direction $\mathbf{v}$ after an interaction. The collision kernel $B(|\mathbf{v}-\mathbf{v}_*|,\cos\theta)$ is a given non-negative function depending on the size of the relative velocity $\mathbf{u} := \mathbf{v}-\mathbf{v}_*$ and $\cos\theta = \frac{\mathbf{u}\cdot\sigma}{|\mathbf{u}|}$, where σ in the $n-1$ dimensional sphere S^{n-1} is referred as the scattering direction of the post-collisional elastic relative velocity.

For the following we will use the velocity elastic (or reversible) interaction law in center of mass-relative velocity coordinates

$$\mathbf{v}' = \mathbf{v} + \frac{1}{2}(|\mathbf{u}|\sigma - \mathbf{u}), \qquad \mathbf{v}'_* = \mathbf{v}_* - \frac{1}{2}(|\mathbf{u}|\sigma - \mathbf{u}) \tag{3}$$
$$B(|\mathbf{u}|, \cos\theta) = |\mathbf{u}|^\lambda b(\cos\theta)\,.$$

We assume that the differential cross section function $b(\cos\theta)$ is integrable with respect to σ on S^{d-1}, referred to as the Grad cut-off assumption, and that it is renormalized such that

$$\int_{S^{d-1}} b(\cos\theta) d\sigma = 1. \tag{4}$$

The parameter λ regulates the collision frequency as a function of the relative speed $|\mathbf{u}|$. This corresponds to the interparticle potentials used in the derivation of the collisional kernel and are referred to as variable hard potentials (VHP) for $0 < \lambda < 1$, hard spheres (HS) for $\lambda = 1$, Maxwell molecules (MM) for $\lambda = 0$, and variable soft potentials (VSP) for $-3 < \lambda < 0$. The $\lambda = -3$ case corresponds to a Coulombic interaction potential between particles. If $b(\cos\theta)$ is independent of σ we call the interactions isotropic (like the case of hard spheres in three dimensions).

Depending on the nature of the collisions, the collision operator can have a number of collision invariants. In the case of the classical Boltzmann equation with elastic collisions, according to the Boltzmann Theorem the only collisional invariants are polynomials of the form $A + B \cdot \mathbf{v} + C|\mathbf{v}|^2$. These give rise to the classical macroscopic conserved quantities

$$\begin{aligned}
\rho(\mathbf{x}, t) &= \int_{\mathbf{v}} f(\mathbf{x}, \mathbf{v}, t) d\mathbf{v} \qquad \text{(density)} \\
\rho(\mathbf{x}, t)\mathbf{V}(\mathbf{x}, t) &= \int_{\mathbf{v}} \mathbf{v} f(\mathbf{x}, \mathbf{v}, t) d\mathbf{v} \qquad \text{(momentum)} \\
\rho(\mathbf{x}, t)e(\mathbf{x}, t) &= \frac{1}{2}\int_{\mathbf{v}} |\mathbf{v}|^2 f(\mathbf{x}, \mathbf{v}, t) d\mathbf{v} \qquad \text{(kinetic energy density)}
\end{aligned} \tag{5}$$

2.1 Boundary conditions

On the spatial boundary $\partial\Omega$ we use a diffusive Maxwell boundary condition which is given by, for $\mathbf{x} \in \partial\Omega$,

$$f(\mathbf{x}, \mathbf{v}, t) = \frac{\sigma_w}{(2\pi R T_w)^(d/2)} \exp\left(-\frac{|\mathbf{v} - \mathbf{V}_w|^2}{2RT_w}\right), \qquad (\mathbf{v} - \mathbf{V}_w) \cdot \mathbf{n} > 0 \tag{6}$$
$$\sigma_w = -\left(\frac{2\pi}{RT_w}\right)^{1/2} \int_{(\mathbf{v}-\mathbf{V}_w)\cdot\mathbf{n}<0} (\mathbf{v} - \mathbf{V}_w) \cdot \mathbf{n} f(\mathbf{x}, \mathbf{v}, t) d\mathbf{v},$$

where $\mathbf{V}_w$ and T_w are the wall velocity and temperature, respectively, and $\mathbf{n}$ is the unit normal vector to the boundary, directed into Ω. The term σ_w accounts for the amount of particles leaving the domain and ensures mass conservation in Ω.

2.2 Spectral formulation

The key step our formulation of the spectral numerical method is the use of the weak form of the Boltzmann collision operator. For a suitably smooth test function $\phi(\mathbf{v})$ the weak form of the collision integral is given by

$$\int_{\mathbb{R}^d} Q(f,f)\phi(\mathbf{v})d\mathbf{v} = \int_{\mathbb{R}^d\times\mathbb{R}^d\times S^{d-1}} f(\mathbf{v})f(\mathbf{v}_*)B(|\mathbf{u}|,\cos\theta)(\phi(\mathbf{v}')-\phi(\mathbf{v}))d\sigma d\mathbf{v}_* d\mathbf{v} \tag{7}$$

If one chooses

$$\phi(\mathbf{v}) = e^{-i\zeta\cdot\mathbf{v}}/(\sqrt{2\pi})^d,$$

then (7) is the Fourier transform of the collision integral with Fourier variable ζ:

$$\begin{aligned}\widehat{Q}(\zeta) &= \frac{1}{(\sqrt{2\pi})^d}\int_{\mathbb{R}^d} Q(f,f)e^{-i\zeta\cdot\mathbf{v}}d\mathbf{v} \\ &= \int_{\mathbb{R}^d\times\mathbb{R}^d\times S^{d-1}} f(\mathbf{v})f(\mathbf{v}_*)\frac{B(|\mathbf{u}|,\cos\theta)}{(\sqrt{2\pi})^d}(e^{-i\zeta\cdot\mathbf{v}'}-e^{-i\zeta\cdot\mathbf{v}})d\sigma d\mathbf{v}_* d\mathbf{v} \\ &= \int_{\mathbb{R}^d} G(\mathbf{u},\zeta)\mathcal{F}[f(\mathbf{v})f(\mathbf{v}-\mathbf{u})](\zeta)d\mathbf{u}, \end{aligned} \tag{8}$$

where $\widehat{[\cdot]} = \mathcal{F}(\cdot)$ denotes the Fourier transform and

$$G(\mathbf{u},\zeta) = |\mathbf{u}|^\lambda \int_{S^{d-1}} b(\cos\theta)\left(e^{-i\frac{\beta}{2}\zeta\cdot|\mathbf{u}|\sigma}e^{i\frac{\beta}{2}\zeta\cdot\mathbf{u}}-1\right)d\sigma \tag{9}$$

Further simplification can be made by writing the Fourier transform inside the integral as a convolution of Fourier transforms:

$$\widehat{Q}(\zeta) = \int_{\mathbb{R}^d}\widehat{G}(\xi,\zeta)\hat{f}(\zeta-\xi)\hat{f}(\xi)d\xi, \tag{10}$$

where the convolution weights $\widehat{G}(\xi,\zeta)$ are given by

$$\widehat{G}(\xi,\zeta) = \frac{1}{(\sqrt{2\pi})^d}\int_{\mathbb{R}^d} G(\mathbf{u},\zeta)e^{-i\xi\cdot u}d\mathbf{u} \tag{11}$$

These convolution weights can be precomputed once to high accuracy and stored for future use. For many collision types the complexity of the integrals in the weight functions can be reduced dramatically through analytical techniques. In this paper we will only consider isotropic scattering in dimension 3 ($b(\cos\theta) = 1/4\pi$). In this case we have that

$$\begin{aligned} G(\mathbf{u},\zeta) &= \frac{|\mathbf{u}|^\lambda}{4\pi}\int_{S^2} e^{-i\frac{\beta}{2}\zeta|\mathbf{u}|\cdot\sigma}e^{i\frac{\beta}{2}\zeta\cdot\mathbf{u}} - 1\ d\sigma \\ &= |\mathbf{u}|^\lambda\left(e^{i\frac{\beta}{2}\zeta\cdot\mathbf{u}}\mathrm{sinc}\left(\frac{\beta|\mathbf{u}||\zeta|}{2}\right)-1\right) \end{aligned} \tag{12}$$

To calculate $\widehat{G}(\xi,\zeta)$, we have

$$\begin{aligned}\widehat{G}(\xi,\zeta) &= \frac{1}{(\sqrt{2\pi})^3}\int_{\mathbb{R}^3} G(\mathbf{u},\zeta)e^{-i\xi\cdot\mathbf{u}}d\mathbf{u} \\ &= \frac{4\pi}{(\sqrt{2\pi})^3}\int_{\mathbb{R}^+} r^{\lambda+2}\left(\operatorname{sinc}\left(\frac{\beta r|\zeta|}{2}\right)\operatorname{sinc}(r|\xi-\frac{\beta}{2}\zeta|)-\operatorname{sinc}(r|\xi|)\right)dr.\end{aligned} \tag{13}$$

This integral will be cut off at a point $r = r_0$, which will be determined below. Given this cutoff point, we can explicitly compute $\widehat{G}$ for integer values of λ.

$$\begin{aligned}\widehat{G}(\xi,\zeta) &= \tfrac{4\pi}{(\sqrt{2\pi})^3} \Big(\frac{q^2(pr_0\sin(pr_0)+\cos(pr_0)-1)-p^2(qr_0\sin(qr_0)+\cos(qr_0)-1)}{\beta|\zeta||\xi-\frac{\beta}{2}\zeta|p^2q^2} \\ &\qquad -\frac{(2-r_0^2|\xi|^2)\cos(r_0|\xi|)+2r_0|\xi|\sin(r_0|\xi|)-2}{|\xi|^4}\Big), \qquad \lambda=1, \\ \widehat{G}(\xi,\zeta) &= \tfrac{4\pi}{(\sqrt{2\pi})^3} \Big(\frac{q\sin(pr_0)-p\sin(qr_0)}{\beta|\zeta||\xi-\frac{\beta}{2}\zeta|pq}-\frac{\sin(|\xi|r_0)-|\xi|r_0\cos(|\xi|r_0)}{|\xi|^3}\Big), \quad \lambda=0.\end{aligned}$$

For other values of λ, this is simply a one-dimensional integral that can be precomputed to high accuracy using numerical quadrature. The entirety of the collisional model being used is encoded in the weights, which gives the algorithm a large degree of flexibility in implementing different models.

3 The Conservative Numerical Method

3.1 Temporal and velocity space discretization

We use an operator splitting method to separate the mechanisms of collisions and advection. The system is split into the subproblems

$$\frac{\partial}{\partial t}f + v\cdot\nabla_{\mathbf{x}}f = 0 \tag{14}$$

$$\frac{\partial}{\partial t}f = Q(f,f), \tag{15}$$

which are solved separately.

Each system is evolved in time using a second-order Runge-Kutta method, and the systems are combined using Strang splitting.

In order to compute the Boltzmann equation we must work on a bounded velocity space, rather than all of $\mathbb{R}^d$. However typical distributions are supported on the entire domain, for example the Maxwellian equilibrium distribution. Even if one begins with a compactly supported initial distribution, each evaluation of the collision operator spreads the support by a factor of $\sqrt{2}$, thus we must use a working definition of an *effective support* of size R for the distribution function. Bobylev and Rjasanow [9] suggested using the temperature of

the distribution function, which typically decreases as $\exp(-|v|^2/2T)$ for large $|v|$, and used a rough estimate of $R \approx 2\sqrt{2}T$ to determine the cutoff. We assume that the distribution function is negligible outside of a ball

$$B_{R_x}(\mathbf{V}(\mathbf{x})) = \{\mathbf{v} \in \mathbb{R}^d : |\mathbf{v} - \mathbf{V}(\mathbf{x})| \leq R_x\}, \tag{16}$$

where $\mathbf{V}(\mathbf{x})$ is the local flow velocity which depends in the spatial variable $\mathbf{x}$. For ease of notation in the following we will work with a ball centered at 0 and choose a length R large enough that $B_{R_x}(\mathbf{V}(\mathbf{x})) \subset B_R(0)$ for all $\mathbf{x}$.

With this assumed support for the distribution f, the integrals in (10) will only be nonzero for $\mathbf{u} \in B_{2R}(0)$. Therefore, we set $L = 2R$ and define the cube

$$C_L = \{\mathbf{v} \in \mathbb{R}^d : |v_j| \leq L,\ j = 1, \ldots, d\} \tag{17}$$

to be the domain of computation. For such domain, the computation of the weight function integral (13) is cut off at $r_0 = L$.

Let $N \in \mathbb{N}$ be the number of points in velocity space in each dimension. Then the uniform velocity mesh size is $\Delta v = \frac{2L}{N}$ and due to the formulation of the discrete Fourier transform the corresponding Fourier space mesh size is given by $\Delta\zeta = \frac{\pi}{L}$.

The mesh points are defined by

$$v_{\mathbf{k}} = \Delta v(\mathbf{k} - N/2)$$
$$\zeta_{\mathbf{k}} = \Delta\zeta(\mathbf{k} - N/2) \tag{18}$$
$$\mathbf{k} = (k_1, \ldots, k_d) \in \mathbb{Z}^d, \qquad 0 \leq k_j \leq N-1,\ \ j = 1, \ldots, d \tag{19}$$

3.2 Collision step discretization

Returning to the spectral formulation (10), the weighted convolution integral then becomes an integral over $-\frac{\pi}{\Delta v} \leq \xi_j \leq \frac{\pi}{\Delta v}$, $j = 1, \ldots, d$.

Similar to what was noted in [24], we can find the cutoff for the integration variable $\mathbf{u}$ through the term

$$g(\mathbf{u}, \mathbf{v}) = f(\mathbf{v})f(\mathbf{v} - \mathbf{u})$$

that appears in the Fourier transform term in (8). As $\operatorname{supp} f = B_L(0)$, we have that

$$\operatorname{supp} g(\mathbf{u}, \cdot) = B_L(0) \cap B_L(\mathbf{u})$$

To simplify notation we will use one index to denote multidimensional sums with respect to an index vector $\mathbf{m}$

$$\sum_{\mathbf{m}=0}^{N-1} = \sum_{m_1, \ldots, m_d = 0}^{N-1}.$$

To compute $\widehat{Q}(\zeta_{\mathbf{k}})$, we first compute the Fourier transform integral giving $\hat{f}(\zeta_k)$ via the FFT. The weighted convolution integral is approximated using the

trapezoidal rule

$$\widehat{Q}(\zeta_{\mathbf{k}}) = \sum_{\mathbf{m}=0}^{N-1} \widehat{G}(\xi_{\mathbf{m}}, \zeta_{\mathbf{k}}) \hat{f}(\xi_{\mathbf{m}}) \hat{f}(\zeta_{\mathbf{k}} - \xi_{\mathbf{m}}) \omega_{\mathbf{m}}, \tag{20}$$

where $\omega_{\mathbf{m}}$ is the quadrature weight and we set $\hat{f}(\zeta_{\mathbf{k}} - \xi_{\mathbf{m}}) = 0$ if $(\zeta_{\mathbf{k}} - \xi_{\mathbf{m}})$ is outside of the domain of integration. We then use the inverse FFT on $\widehat{Q}$ to calculate the integral returning the result to velocity space.

Note that in this formulation the distribution function is not periodized, as is done in the collocation approach of Pareschi and Russo [35]. This is reflected in the omission of Fourier terms outside of the Fourier domain. All integrals are computed directly only using the FFT as a tool for faster computation.The convolution integral is accurate to at least the order of the quadrature. The calculations below use the trapezoid rule, but in principle Simpson's rule or some other uniform grid quadrature can be used. However, it is known that the trapezoid rule is spectrally accurate for periodic functions on periodic domains (which is the basis of spectral accuracy for the FFT), and the same arguments can apply to functions with sufficient decay at the integration boundaries [4]. These accuracy considerations will be investigated in future work. The overall cost of this step is $O(N^{2d})$.

3.3 Discrete conservation enforcement

This implementation of the collision mechanism does not conserve all of the quantities of the collision operator. To correct this fact, we formulate these conservation properties as a constrained optimization problem as proposed in [21, 22]. Depending on the type of collisions we can change this constraint set (for example, inelastic collisions do not preserve energy). We focus here just on the case of elastic collisions, which preserve mass, momentum, and energy.

Let $M = N^d$ be the total number of grid points, let $\tilde{\mathbf{Q}} = (\tilde{Q}_1, \ldots, \tilde{Q}_M)^T$ be the result of the spectral formulation from the previous section, written in vector form, and let ω_j be the quadrature weights over the domain in this ordering. Define the integration matrix

$$\mathbf{C}_{5\times M} = \begin{pmatrix} \omega_j \\ v_j^i \omega_j \\ |\mathbf{v}_j|^2 \omega_j \end{pmatrix},$$

where v^i, $i = 1, 2, 3$ refers to the ith component of the velocity vector. Using this notation, the conservation method can be written as a constrained optimization problem.

$$\text{Find } \mathbf{Q} = (Q_1, \ldots, Q_M)^T \text{ that minimizes } \frac{1}{2}\|\tilde{\mathbf{Q}} - \mathbf{Q}\|_2^2 \text{ such that } \mathbf{CQ} = \mathbf{0} \tag{21}$$

The solution is given by

$$\begin{aligned}\mathbf{Q} &= \tilde{\mathbf{Q}} + \mathbf{C}(\mathbf{C}\mathbf{C}^T)^{-1}\mathbf{C}\tilde{\mathbf{Q}} \\ &:= \mathbf{P}_N\tilde{\mathbf{Q}}\end{aligned} \tag{22}$$

Overall the collision step in semi-discrete form is given by

$$\frac{\partial \mathbf{f}}{\partial t} = \mathbf{P}_N\tilde{\mathbf{Q}} \tag{23}$$

The overall cost of the conservation portion of the algorithm is a $O(N^d)$ matrix-vector multiply, significantly less than the computation of the weighted convolution.

3.4 Spatial and Transport discretization

For simplicity this will be presented in 1D in space, though the ideas apply to higher dimensions. In this case the transport equation reduces to

$$\frac{\partial f}{\partial t}(x, \mathbf{v}, t) + v_1 \frac{\partial}{\partial x} f(x, \mathbf{v}, t) = 0.$$

We partition the domain into cells of size Δx_j (not necessarily uniform) with cell centers x_j. Using a finite volume approach, we integrate the transport equation over a single cell to obtain

$$\frac{f_j^{n+1}(v_i) - f_j^n(v_i)}{\Delta t} + \frac{F_{j+1/2}^n - F_{j-1/2}^n}{\Delta x_j} = 0, \tag{24}$$

where $t^n = n\Delta t$ and $F_{j\pm 1/2}^n$ is an approximation of the edge fluxes $v_1 f$ of the cell between time t^n and t^{n+1}. We use a second order upwind scheme defined by

$$F_{j+1/2}^n = \begin{cases} v_1(f_j^n + \frac{\Delta x}{2}\sigma_j^n), & v_1 \geq 0 \\ v_1(f_{j+1}^n - \frac{\Delta x}{2}\sigma_{j+1}^n), & \text{otherwise,} \end{cases} \tag{25}$$

where σ_j is a cell slope term used in the reconstruction defined by the minmod limiter.

$$\sigma_j = \text{minmod}\left(\frac{f_{j+1}^n - f_j^n}{x_{j+1} - x_j}, \frac{f_j^n - f_{j-1}^n}{x_j - x_{j-1}}, \frac{f_{j+1}^n - f_{j-1}^n}{x_{j+1} - x_{j-1}}\right) \tag{26}$$

For reconstructions at the boundary of the physical domain, ghost cells and extrapolation are used to determine the reconstructed slope [27].

On wall boundaries the incoming flux is determined using the diffusive reflection formula (6). For problems without meaningful boundary interactions (e.g. shocks), a no-flux boundary condition is applied for the incoming characteristics.

4 Parallelization

4.1 Shared Memory Parallelization

The most computationally expensive term to evaluate in one time step of the Boltzmann solver is the weighted convolution in $\hat{Q}$. This requires computing the sum of N^3 terms: $\sum_{\mathbf{m}=0}^{N-1} \widehat{G}(\xi_{\mathbf{m}}, \zeta_{\mathbf{k}}) \hat{f}(\xi_{\mathbf{m}}) \hat{f}(\zeta_{\mathbf{k}} - \xi_{\mathbf{m}}) \omega_{\mathbf{m}}$ for each of the N^3 values of $\zeta_{\mathbf{k}}$ on the numerical grid. Each sum draws from essentially the same data and recombines it in a different way, which makes it ideal for shared memory parallelization.

Shared memory parallelization refers to utilizing an architecture in which multiple processes can simultaneously access the same address space in memory. This type of parallelization has grown more prominent over the past decade as multicore processors are becoming more and more prevalent in typical personal computers, and they are the building blocks of larger scale high performance computing systems. The primary motivation for this type of architecture is that memory access speed is much less than the speed of a processor, and having a common pool of memory mitigates the need for memory transfers between processors of the system. In this architecture, computational work is divided into a discrete number of threads that are distributed to the available processors, which compute each thread separately. Logistically, the most difficult hurdles to overcome in this computing paradigm are race conditions, where one thread may not recieve the correct data depending on whether another thread has modified it before access, load balancing, where work is not distributed equally among the threads, and blocking, where one thread must wait on another thread to complete some work before it can begin.

We utilize the OpenMP API [32] to parallelize the weighted convolution. This interface is available on most computing architectures and compilers, and in many cases can give a significant speedup for only adding two or three simple lines of code. Typically the same source code can be executed on anything ranging from a personal laptop to a high performance computing node, which illustrates the great portability of this model.

For our application, the hurdles associated with parallelization mentioned above are not a problem. We avoid load balancing problems by evenly subdividing the work among the available cores, and each convolution is expected to take the same amount of operations as any other. Each thread makes a computation based the weight $\widehat{G}$ and distribution function $\hat{f}$, and is not dependent on communication with other threads, so there is no need to worry about race conditions or thread blocking. Taking this into account the speedup would be expected to scale linearly with p, the number of cores that can access the memory, however memory access between the shared memory units in the node still present latency issues on the total speedup, as observed in the test below.

In Tables 1–4 we present the timing results of a single evaluation of the collision operator for $N = 16$ and $N = 24$ points in each velocity direction. These tests are performed on the Texas Advanced Computing Center's supercomputers Ranger and Stampede. On Ranger, a single computational node contains

four AMD Opeteron quad-core 64 bit processors, for a total of 16 cores with a frequency of 2.3 GHz each, 32 GB of shared memory, and a peak performance of 128 GFLOPS per node. Stampede is TACC's newest system, officially deploying in January 2013 and uses the new Intel MIC architecture. Each computational node consists of two Intel Xeon E-5 Processors and a Xeon Phi Coprocessor (the MIC component). The E-5 processors have 8 cores each at 2.7 GHz and 32 GB of memory, for a total of 16 cores, and the Phi Coprocessor contains 61 cores at 1.1 GHz and 8GB of fast-access memory for a total of $\sim$1070 GFLOPS of performance by itself. The peak performance of the entire system was benchmarked at 3959 Teraflops in November 2012, making it the 7th fastest supercomputer in the world [1], and has more components yet to be added. As Stampede is still in its trial period the necessary libraries have not been created to run this code on the coprocessor, so the following results are run only on the 16 cores of the E-5 processors.

cores	time (s)	ratio	total speedup
1	0.068		
2	0.044	1.54	1.54
4	0.027	1.63	2.52
8	0.019	1.42	3.58
16	0.018	1.05	3.78

Table 1: Time for single evaluation of collision operator on one node with OpenMP. Ranger with N=16

cores	time (s)	ratio	total speedup
1	0.90567		
2	0.4899	1.85	1.85
4	0.3021	1.62	3.0
8	0.19197	1.57	4.72
16	0.1769	1.09	5.12

Table 2: Time for single evaluation of collision operator on one node with OpenMP. Ranger with N=24

cores	time (s)	ratio	total speedup
1	0.03337		
2	0.01627	2.05	2.05
4	0.00976	1.67	3.41
8	0.00599	1.63	5.57
16	0.00408	1.47	8.18

Table 3: Time for single evaluation of collision operator on one node with OpenMP. Stampede with N=16

cores	time (s)	ratio	total speedup
1	0.34386		
2	0.17446	1.97	1.97
4	0.103	1.69	3.34
8	0.06107	1.68	5.63
16	0.04611	1.32	7.46

Table 4: Time for single evaluation of collision operator on one node with OpenMP. Stampede with N=24

4.2 Distributed Memory Parallelization

The total number of operations required to compute the all of the collisional terms in a single timestep is $O(MN^6)$, where M is the total number of physical space grid points. When dividing the computational work, in shared memory parallelization we are restricted to the number of processors that can physically access the shared memory. Distributed memory parallelization on the other hand consists of starting multiple processes that can communicate with each other, each with their own private address space.

The main drawback of distributed memory is memory access time. Unlike shared memory, distributed memory is less local and thus requires much more time to receive information from a process that is not in its address space when compared to a similar shared memory access. Therefore, it is important to design algorithms that limit communication between processes as much as possible. Furthermore, many of the same issues from shared memory programming still apply, such as load balancing and blocking.

Recall that the collision term in the Boltzmann equation is local in space, thus each grid point in physical space only requires $\hat{f}(\mathbf{x}, \cdot)$ and the precomputed weights $\widehat{G}$ to evaluate the collision term. This allows for a natural decomposition of the computational domain by separating across physical grid points. The only communication between the computational nodes is in the transport term (24). Each process simply sends and receives the $O(N^3)$ values of $\hat{f}(\mathbf{x}, \zeta)$ at the edges of the decomposed domain required to calculate the spatial flux. The communication between processes is managed by the Message Passing Interface (MPI) protocol [17] using an interleaved ghost cells technique [23]. This mitigates the possibility of message deadlock by ensuring that all even indexed processes send or receive data at the same time, while the odd indexed processors receive data from or send data to the even processors, respectively. The code below fills the edge cells on nodes to either side. Extrapolation is used to fill ghost cells at the physical domain boundaries, which is removed from the code below for clarity of presentation.

```
---------------------------------
if((rank % 2) == 0) { //EVEN NODES
   MPI_Ssend(f[nX_node+1],N*N*N,MPI_DOUBLE,rank+1,0,MPI_COMM_WORLD);
```

```
    MPI_Recv(f[1],N*N*N,MPI_DOUBLE,rank-1,0,MPI_COMM_WORLD, &status);

    MPI_Ssend(f[nX_node],N*N*N,MPI_DOUBLE,rank+1,0,MPI_COMM_WORLD);

    MPI_Recv(f[0],N*N*N,MPI_DOUBLE,rank-1,0,MPI_COMM_WORLD, &status);

    MPI_Ssend(f[2],N*N*N,MPI_DOUBLE,rank-1,1,MPI_COMM_WORLD);

    MPI_Recv(f[nX_node+2],N*N*N,MPI_DOUBLE,rank+1,1,MPI_COMM_WORLD, &status);

    MPI_Ssend(f[3],N*N*N,MPI_DOUBLE,rank-1,1,MPI_COMM_WORLD);

    MPI_Recv(f[nX_node+3],N*N*N,MPI_DOUBLE,rank+1,1,MPI_COMM_WORLD, &status);
  }
  else { //ODD NODES
    MPI_Recv(f[1],N*N*N,MPI_DOUBLE,rank-1,0,MPI_COMM_WORLD, &status);

    MPI_Ssend(f[nX_node+1],N*N*N,MPI_DOUBLE,rank+1,0,MPI_COMM_WORLD);

    MPI_Recv(f[0],N*N*N,MPI_DOUBLE,rank-1,0,MPI_COMM_WORLD, &status);

    MPI_Ssend(f[nX_node],N*N*N,MPI_DOUBLE,rank+1,0,MPI_COMM_WORLD);

    MPI_Recv(f[nX_node+2],N*N*N,MPI_DOUBLE,rank+1,1,MPI_COMM_WORLD, &status);

    MPI_Ssend(f[2],N*N*N,MPI_DOUBLE,rank-1,1,MPI_COMM_WORLD);

    MPI_Recv(f[nX_node+3],N*N*N,MPI_DOUBLE,rank+1,1,MPI_COMM_WORLD, &status);

    MPI_Ssend(f[3],N*N*N,MPI_DOUBLE,rank-1,1,MPI_COMM_WORLD);
  }
  ---------------------------------
```

To make a rough estimate of the total speedup let n be the number of processes and np be the number of cores used (assume a fixed number of cores per node). Then the speedup can be described by, to leading order,

$$\frac{T_{\rm SERIAL}}{T_{\rm PARALLEL}} = \frac{CMN^6 T_{\rm FLOP}}{4nN^3 T_{\rm MEM} + CMN^6 T_{\rm FLOP}/np}, \tag{27}$$

where $T_{\rm MEM}$ is the average time to transfer a double precision value and $T_{\rm FLOP}$ is the time for a single floating point operation. As n becomes large with N and M fixed, more and more data transfer is required, however one would need $4n^2 p T_{MEM}/T_{FLOP} \approx CMN^3$ before the memory access time would begin to dominate the collisional computations.

We remark that this parallelization approach can work for other collisional models that take the form of a weighted convolution in Fourier space, for ex-

ample collisional models with anisotropic scattering or the Landau equation [18, 20].

We test the this method with the sudden change in wall temperature example suggested by Aoki et al. [3]. These authors computed this example using the BGK approximation of the collision operator and finite differences. This was later computed in [22] with a preliminary first order serial version of this conservative spectral method for the fullt nonlinear Boltzmann operator. In this problem the gas is assumed to be initially at equilibrium, and the temperature of the wall at the boundary of the domain is instantaneously changed at $t = 0$. This gives rise to a discontinuous distribution function at the wall, which propagates into the domain and eventually forms a shock. This problem is difficult to compute with a Monte Carlo method due a distribution function that is far from equilibrium near the wall and the fact that the the shock develops over a long period of time.

In Figure 4.2 we show shock formation due to a sudden change in wall temperature. Unlike the computations in [3], only two grid points are used per mean free path in the interior of the domain. Near the wall, the grid is refined to eight points per mean free path to better capture the finer dynamics where the distribution function is discontinuous. In both examples, we set the scaled Knudsen number to $\varepsilon = 1$. Note that despite the discontinuity we do not observe any Gibbs phenomena in the solution. We hypothesize that this is due to the mixing effects of the convolution weights; this will be explored in future work.

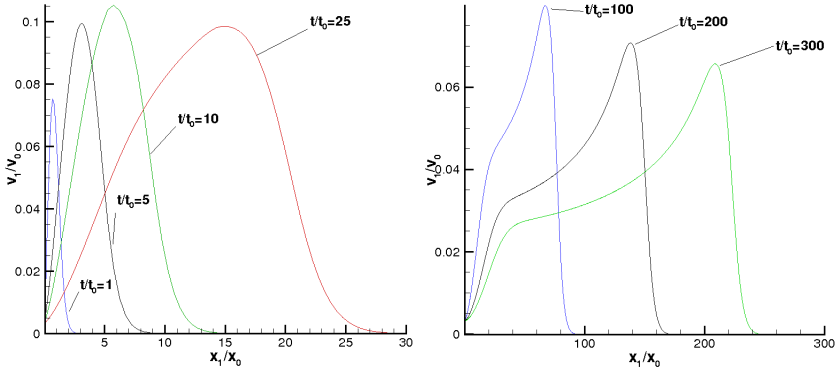

Figure 1: Formation of shock from sudden heating of wall. Evolution of bulk velocity.

In Table 5 we show the wall time scaling results for the sudden heating example. We see a near perfect linear scaling as more nodes are added. Overall the openMP/MPI hybrid scheme gives a speedup of $\sim 4n$, where n is the total number of nodes. Based on the openMP results above, we would expect a $8n$ increase on the Stampede processors. For the computations above, we used 32 nodes, or 512 cores, to obtain a speedup of ~ 128 compared to the computations

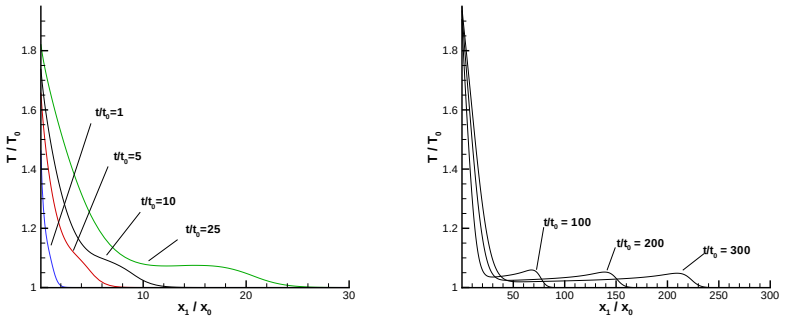

Figure 2: Formation of shock from sudden heating of wall. Evolution of kinetic temperature.

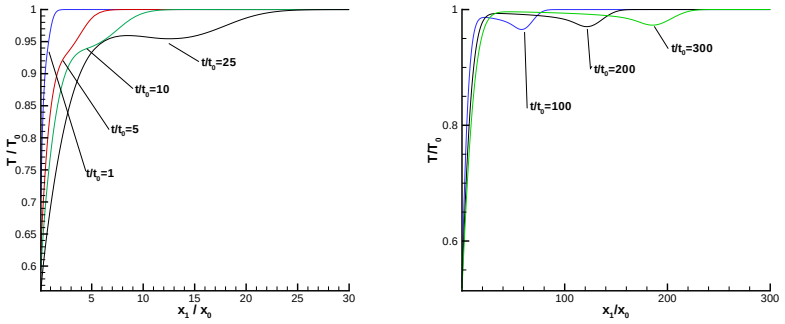

Figure 3: Results from sudden cooling of wall. Evolution of kinetic temperature.

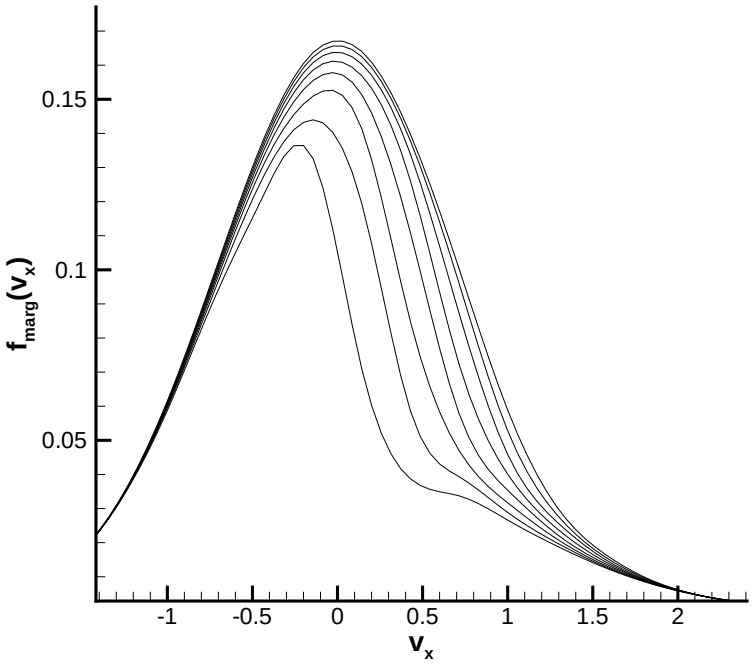

Figure 4: Sudden heating: evolution of discontinuous marginal distribution near the wall. The plots from bottom to top are the marginal distribution of f in eight equispaced cells on $x = [0, 1]$

in [22].

nodes	cores	time (s)
1	16	456.313
2	32	235.315
4	64	120.762
8	128	61.345
16	256	30.943
32	512	15.252
64	1024	7.813
128	2048	4.042

Table 5: Computational time for a single timestep in sudden heating example from Figure 1.

5 Conclusions

We have extended the spectral method of Gamba and Tharkabhushanam to a second order scheme with a nonuniform grid in physical space, and investigated its scaling to high performance computing. The method showed nearly linear speedup across nodes when applied to a large computation problem solved on a supercomputer. However, at some point memory access will still become a problem, not with transfer between nodes but with memory access on a single node. The most expensive object to store is the six dimensional weight array $\widehat{G}(\zeta, \xi)$, and if N becomes too large it may not fit on a single node's memory, significantly slowing down computation. At that point, it may be faster to simply compute the weights on the fly, as flops are much cheaper than memory accesses on the large distributed systems. For the isotropic scattering cross sections considered in this paper this would require computation of at most a one dimensional integral for each velocity grid point, and in the hard sphere and Maxwell molecules we have an exact formula to find the weights. Memory management will become especially important when implementing the code on MICs, which have much more processing power but relatively smaller memory. This will be a subject of future work.

6 Acknowledgments

Thanks to Irene Gamba for introducing me to this method, and thanks to Andrew Christlieb for opening my eyes to parallel programming. This work has been supported by the NSF under grant number DMS-0636586.

References

[1] `http://www.top500.org/lists/2012/11/`.

[2] R. Alonso, I. M. Gamba, and S. H. Tharkabhushanam, *Accuracy and consistency of Lagrangian based conservative spectral method for space-homogeneous Boltzmann equation.* submitted, 2012.

[3] K. Aoki, Y. Sone, K. Nishini, and H. Sugimoto, *Numerical analysis of unsteady motion of a rarefied gas caused by sudden changes of wall temperature with special interest in the propogation of a discontinuity in the velocity distribution function*, in Rarefied Gas Dynamics, A. E. Beylich, ed., VCH, Weinheim, 1991, pp. 222–231.

[4] K. Atkinson, *An Introduction to Numerical Analysis*, John Wiley and Sons, New York, 1984.

[5] G. A. Bird, *Molecular Gas Dynamics*, Clarendon Press, Oxford, 1994.

[6] A. V. Bobylev, *The theory of the nonlinear spatially uniform Boltzmann equation for Maxwell molecules.*, Mathematical Physics Reviews, 7 (1988), pp. 111–233. Soviet Sci. Rev. Sect. C Math. Phys.Rev., 7, Harwood Academic Publ., Chur.

[7] A. V. Bobylev, A. Palczewski, and J. Schneider, *A consistency result for a discrete velocity model of the Boltzmann equation*, SIAM J. Numer. Anal., 5 (1997), pp. 1865–1883.

[8] A. V. Bobylev and S. Rjasanow, *Difference scheme for the Boltzmann equation based on the fast Fourier transform*, Euro. J. Mech. B Fluids, 16 (1997), pp. 293–306.

[9] ———, *Fast deterministic method of solving the Boltzmann equation for hard spheres*, Euro. J. Mech. B Fluids, 18 (1999), pp. 869–887.

[10] J. E. Broadwell, *Study of rarefied shear flow by the discrete velocity method*, J. Fluid Mech., 19 (1964), pp. 404–414.

[11] J. E. Broadwell, D. Goldstein, and B. Sturtevant, *Investigation of the motion of discrete velocity gases*, no. 118 in Rar. Gas Dynam., Progress in Astronautics e Aeronautics, AIAA, New York, 1989.

[12] C. Buet, *A discrete velocity scheme for the Boltzmann operator of rarefied gas dynamics*, Transp. Theo. Stat. Phys., 25 (1996), pp. 33–60.

[13] Y. Cheng, I. M. Gamba, A. Majorana, and C.-W. Shu, *Performance of a discontinuous Galerkin solver for semiconductor Boltzmann equations*, in Proccedings of the 14th International Workshop on Computational Electronics, 2010.

[14] A. Frezzotti, G. Ghiroldi, and L. Gibelli, *Direct solution of the Boltzmann equation for a binary mixture on GPUs*, in Proceedings of the 27th International Symposium on Rarefied Gas Dynamics, 2011, pp. 884–889.

[15] ——, *Solving model kinetic equations on GPUs*, Computers and Fluids, 50 (2011), pp. 136–146.

[16] ——, *Solving the Boltzmann equation on GPUs*, Comput. Phys. Comm., 182 (2011), pp. 2445–2453.

[17] E. Gabriel, G. E. Fagg, G. Bosilca, T. Angskun, J. J. Dongarra, J. M. Squyres, V. Sahay, P. Kambadur, B. Barrett, A. Lumsdaine, R. H. Castain, D. J. Daniel, R. L. Graham, and T. S. Woodall, *Open MPI: Goals, concept, and design of a next generation MPI implementation*, in Proceedings, 11th European PVM/MPI Users' Group Meeting, Budapest, Hungary, 2004.

[18] I. Gamba and J. Haack, *Conservative deterministic spectral Boltzmann solver near the grazing collisions limit*, in Proceedings of the 28th International Symposium on Rarefied Gas Dynamcis, 2012, pp. 326–333.

[19] ——, *High performance computing with a conservative spectral Boltzmann solver*, in Proceedings of the 28th International Symposium on Rarefied Gas Dynamcis, 2012, pp. 334–341.

[20] I. M. Gamba and J. Haack, *A conservative spectral method for the Boltzmann equation with anisotropic scattering and the grazing collisions limit*. submitted, 2013.

[21] I. M. Gamba and S. H. Tharkabhushanam, *Spectral - Lagrangian based methods applied to computation of non-equilibrium statistical states*, J. Comput. Phys, 228 (2009), pp. 2016–2036.

[22] ——, *Shock and boundary structure formation by spectral-Lagrangian methods for the inhomogeneous Boltzmann transport equation*, J. Comput. Math, 28 (2010), pp. 430–460.

[23] W. Gropp, E. Lusk, and A. Skjellum, *Using MPI : portable parallel programming with the message-passing interface*, MIT Press, 1999.

[24] I. Ibragimov and S. Rjasanow, *Numerical solution of the Boltzmann equation on the uniform grid*, Computing, 69 (2002), pp. 163–186.

[25] T. Inamuro and B. Sturtevant, *Numerical study of discrete velocity gases*, Phys. Fluids A, 2 (1990), pp. 2196–2203.

[26] A. Kashkovsky, A. Shersnev, and M. Ivanov, *Efficient CUDA implementation in the DSMC method*, in Proceedings of the 28th International Symposium on Rarefied Gas Dynamics, 2012, pp. 511–518.

[27] R. Leveque, *Finite Volume Methods for Hyperbolic Problems*, Cambridge University Press, Cambridge, 2002.

[28] E. Malkov and M. Ivanov, *Parallelization of algorithms for solving the boltzmann equation for GPU-based computations*, in Proceedings of the 27th International Symposium on Rarefied Gas Dynamics, 2011, pp. 946–951.

[29] E. Malkov, S. Poleshkin, and M. Ivanov, *Discrete velocity scheme for solving the boltzmann equation with the GPGPU*, in Proceedings of the 28th International Symposium on Rarefied Gas Dynamics, 2012, pp. 318–325.

[30] A. Morris, P. L. Varghese, and D. Goldstein, *Monte Carlo solution of the Boltzmann Equation via a discrete velocity model*, J. Comp. Phys., 230 (2011), pp. 1265–1280.

[31] K. Nanbu, *Direct simulation scheme derived from the Boltzmann equation in monocomponent gases*, J. Phys. Soc. Japan, 52 (1983), pp. 2042–2049.

[32] OpenMP Architecture Review Board, *OpenMP application program interface version 3.0*, May 2008.

[33] V. Panferov and A. Heintz, *A new discrete velocity method for the Boltzmann equation*, Math. Meth. Appl. Sci., 25 (2002), pp. 571–593.

[34] L. Pareschi and B. Perthame, *A Fourier spectral method for homogeneous Boltzmann equations*, Trans. Theo. Stat. Phys., 25 (1996), pp. 369–382.

[35] L. Pareschi and G. Russo, *Numerical solution of the Boltzmann equation I: Spectrally accurate approximation of the collision operator*, SIAM J. Num. Anal., (2000), pp. 1217–1245.

[36] F. Rogier and J. Schneider, *A direct method for solving the Boltzmann equation*, Transp. Theo. Stat. Phys., 23 (1994), pp. 313–338.